ArcMap .CAL Script: Generate Radial Polygons

Contributed by Bert Granberg
16, Feb. 2009
Last Updated 16, Feb. 2009

This ArcMap field calculator script will populate the SHAPE geometry field in an ArcGIS polygon feature class from basic radial attributes used in FCC ULS radio transmission data, including mobile phones.The ULS data is freely downloadable from the FCC ULS site.

The FCC license requires eight radials at specifed azimuths (0,45,90,135,180,225,270,& 315 degrees) be provided by for each proposed location. A radial is a pizza-shaped wedge of geographic space that roughly represents the service area boundary of an antenna by factoring in height, terrain, and atmosphere. An obvious next step would be using a elevation model dataset to better analyze the impact of terrain.

This script requires four input values stored in attribute fields (you need these values in a existing polygon feature class with empty geometries):
- The x coordinate for the center point (usually a tower)
- The y coordinate for the center point (usually a tower)
- The distance to the service area boundary (in meters)
- The radial's azimuth measured in degrees (again, these need to be in increments of 45 degrees starting with 0

Use within an edit session is recommended.

TO USE:
- define an empty geodatabase or shapefile polygon feature class with at least the required x,y,distance, and radius azimuth fields(the curves will be converted to line segments with shapefile format)
- load the tabular data from the FCC ULS radial tables into this polygon feature class

- open the attribute table for this polygon feature class and select the SHAPE field
- then hit CTRL + SHIFT + F to open the field calculator.

- In the field calculator, check the Advanced Option and paste this script in the Pre-Logic VBS Script Code text box.
- Make sure that the four input field names in the script correspond to the names of your fields containg these values
- Type the word pSegmentCollection into the bottom text box and hit OK

Note: Input data should look something like this before running the script:

```
'ArcMap 9.3 .CAL script

Dim originX As Double
Dim originY As Double
Dim distance As Double
Dim sector As Integer

'*** Required Inputs from Field Calculator Fields List
originX = [x] 'x coordinate for tower location, in some linear unit, not degrees
originY = [y] 'y coordinate for tower location, in some linear unit, not degrees
distance = [Distance] 'sector distance from tower location
sector = [Azimuth] 'sector angle in degrees, must be 0,45,90,135,180,225,270,or 315


Dim shortSideOffset As Double 'short side offset
Dim longSideOffset As Double 'long side offset

Dim pointAX As Double 'x coordinate for Point A, left point looking from origin
Dim pointAY As Double 'y coordinate for Point A, left point looking from origin
Dim pointBX As Double 'x coordinate for Point B, right point looking from origin
Dim pointBY As Double 'y coordinate for Point B, right point looking from origin
```

```
Dim pSegmentCollection As ISegmentCollection
Dim pLine_O_To_A As ILine
Dim pLine_B_To_O As ILine
Dim pCirArc_A_To_B As ICircularArc
Dim pPointO As IPoint
Dim pPointA As IPoint
Dim pPointB As IPoint

'make point at origin
Set pPointO = New Point
pPointO.PutCoords originX, originY

'The FCC ULS sector radius boundaries are all 22.5 degrees from an x or y axis
'sin and cos functions for 22.5 degrees provide the offsets in the x and y directions for all radius endpoints
shortSideOffset = Abs(Sin(22.5 * 0.0174532925) * distance) 'constant converts radians to degrees
longSideOffset = Abs(Cos(22.5 * 0.0174532925) * distance) 'constant converts radians to degrees

Select Case sector

   Case 0 'North

      pointAX = originX - shortSideOffset
      pointAY = originY + longSideOffset

      pointBX = originX + shortSideOffset
      pointBY = originY + longSideOffset

   Case 45 'NE

      pointAX = originX + shortSideOffset
      pointAY = originY + longSideOffset

      pointBX = originX + longSideOffset
      pointBY = originY + shortSideOffset

   Case 90  'East

      pointAX = originX + longSideOffset
      pointAY = originY + shortSideOffset

      pointBX = originX + longSideOffset
      pointBY = originY - shortSideOffset

   Case 135  'SE

      pointAX = originX + longSideOffset
      pointAY = originY - shortSideOffset

      pointBX = originX + shortSideOffset
      pointBY = originY - longSideOffset

   Case 180  'South

      pointAX = originX + shortSideOffset
      pointAY = originY - longSideOffset

      pointBX = originX - shortSideOffset
      pointBY = originY - longSideOffset

   Case 225 'SW

      pointAX = originX - shortSideOffset
      pointAY = originY - longSideOffset

      pointBX = originX - longSideOffset
```

```
        pointBY = originY - shortSideOffset


    Case 270 'West

        pointAX = originX - longSideOffset
        pointAY = originY - shortSideOffset

        pointBX = originX - longSideOffset
        pointBY = originY + shortSideOffset

    Case 315 'NW

        pointAX = originX - longSideOffset
        pointAY = originY + shortSideOffset

        pointBX = originX - shortSideOffset
        pointBY = originY + longSideOffset
End Select

Set pPointA = New Point
Set pPointB = New Point

pPointA.PutCoords pointAX, pointAY
pPointB.PutCoords pointBX, pointBY

Set pLine_O_To_A = New Line
Set pCirArc_A_To_B = New CircularArc
Set pLine_B_To_O = New Line

pLine_O_To_A.PutCoords pPointO, pPointA
pCirArc_A_To_B.PutCoords pPointO, pPointA, pPointB, esriArcClockwise
pLine_B_To_O.PutCoords pPointB, pPointO

Set pSegmentCollection = New Polygon
pSegmentCollection.AddSegment pLine_O_To_A
pSegmentCollection.AddSegment pCirArc_A_To_B
pSegmentCollection.AddSegment pLine_B_To_O
```